CG summary

Simone Guggiari

1.2 Graphics pipeline

in: Geometry, Materials, Lights, camera → GP → out: display colors

BRDF (6D fz) bidir. refl distrib fz → use models (materials)

vx: modeling trans (obj → world) → viewing trans (world → cam) → 3D clip (frustum: • reduce • stability) → light shading texturing (compute color (light, mat, normals)) → px: proj to screen sp (3D → 2D) → scan conversion (interpolate vx → px) (discretization of primitives)

Nyquist theorem → sample-tane max frequency
has infinite frequency → aliasing

Programmer's view

CPU → vx op: Vertex Processing (• transforms • lighting • flow control) → rasterization → px op: Fragment processing (• shading • texturing • blending) → Display

vx shader → interpol → frag shader (programmable)
attributes /vx → varying /vx → varying /frag → color /frag

attributes → VS VS VS → varying interpolate varying → FS → color

Code
- vx shader
- frag shader

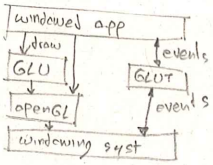
Fragment ≠ pixel (frag ⊠) contains • position • depth • texture coord ..

G API access graphics HW independently, abstract complexity
low lvl openGL, directX (drawing commands)
high lvl Renderman, .. (DAG for scene)

open GL standardized graphics lib
similar to C
updated for HW
language bindings

double buffer (one render, one build)

Code
- example application (display fz, VS, FS)

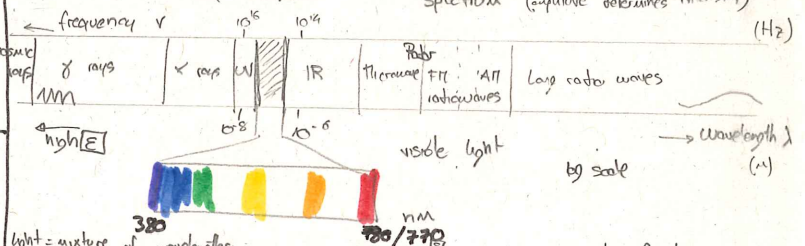


libraries
gl.lib ! main
glu.lib ! utilities
glut.lib ! toolkit
glext.h ! extensions

primitives
glDrawElements (X, count, uint, 0);
GL_POINTS
LINES
POLYGON
TRIANGLES
QUADS
TRIANGLE_STRIP
FAN

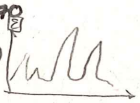
2.1 LIGHT AND COLORS

what is light = EM radiation — spectrum (amplitude determines intensity)



light = mixture of wavelengths

SPD = spectral power distribution — color = function



each ray carries P(λ) → ∞ dim vector (humans can't process)
↳ project into subspace (discretize using cones) — some info lost (different SPD might look same to us)

$x = x_1 n_1 + x_2 n_2 + x_3 n_3$

$x = (x_1 \cdot n_1) n_1 + \dots$

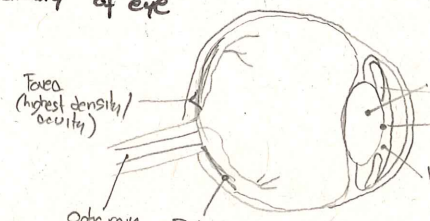
$x^P = (\quad) + (\quad)$ ——— (leave term out for projection)

$x^{3D} = (x_1\ x_2\ x_3) \rightarrow x^\infty = x(\lambda)$

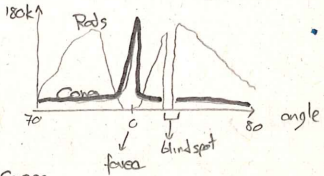
$x^P(\lambda) = x_1 n_1(\lambda) + x_2 n_2(\lambda) + x_3 n_3(\lambda)$
(combination of primary colors)

coord are continuous inner prod
$x_i = \int x(\lambda) n_i(\lambda) d\lambda$

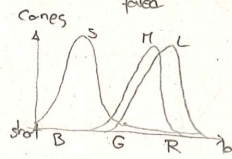
Anatomy of eye


Fovea (highest density/ acuity)
Lens (focus light on retina) → muscle allows it (reshape lens)
Pupil = hole in iris
Iris (lets light into eye) controls & dilates to light
Optic nerve (signals to brain)
Retina (react to light) cells

Cones: concents. fovea, color (3 kind) | high acuity
Rods: lower acuity (require less light) respond to intensity only

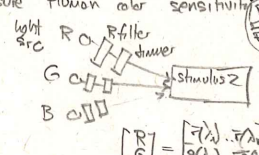


Animals: 2D/1D subspaces
mammals / dolphins.
8D



3 basis vectors/cones for projection in 3D

CIE Primary system (1931) commission internationale de l'éclairage
measure human color sensitivity: 435.8 (B) 546.1 (G) 700 (R) nm

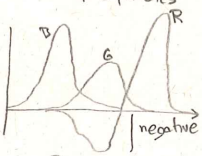


(shutter controls intensity)
(control shutter to have same color)

$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} \bar{r}(\lambda_1) \dots \bar{r}(\lambda_n) \\ \bar{g}(\lambda_1) \dots \bar{g}(\lambda_n) \\ \bar{b}(\lambda_1) \dots \bar{b}(\lambda_n) \end{bmatrix} \begin{bmatrix} P(\lambda_1) \\ P(\lambda_2) \\ P(\lambda_n) \end{bmatrix}$

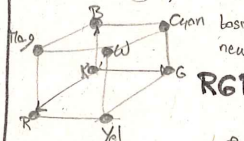
color matching as matrix mult
intensity for 3 primaries — color matching functions — SPD of test light

(use monochromatic test light $\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$)



⇒ add red to ref light

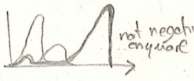
negative matching value → some colors cannot be written as lin-comb



Cyan basis is arbitrary → perceptual system dependent
new basis → linear transf → same space

RGB (additive)

CIE XYZ $X = \int_0^\infty P(\lambda) \bar{x} d\lambda$ (save Y, Z)

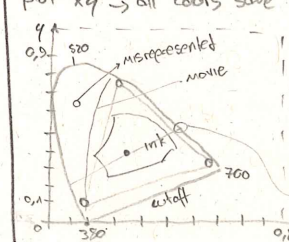


(X,Y,Z) quantifies any P(λ) stimulus

$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 2.36 & -0.515 & 0.005 \\ -.89 & 1.426 & 0.014 \\ -0.46 & 0.088 & 1.009 \end{pmatrix} \cdot \begin{pmatrix} \bar{r}(\lambda) \\ \bar{g}(\lambda) \\ \bar{b}(\lambda) \end{pmatrix}$

Chromacity (x,y), Y = brightness
$x = \frac{X}{X+Y+Z}$ $y = \frac{Y}{X+Y+Z}$ $Z = 1-x-y$

plot xy → all colors save brightness

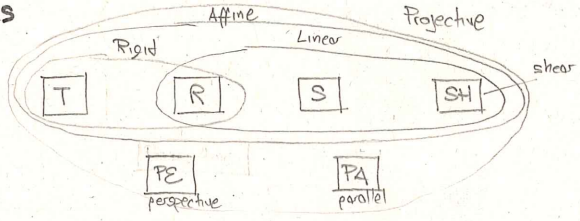


- Primaries along curved boundary (spectral colors) single wavelength
- linear combin: line below 2 pts
- Gamut: lin comb of 3 pts
- ⊙ = white pt @ (0.34, 0.34)
- → dominant wavelength
- → inverse color

non-spectral purples (contain multiple wavelengths)

2.2 TRANSFORMATIONS

Transformations

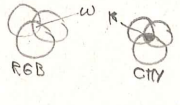


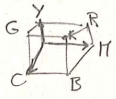
* other color spaces

CMY (passive color systems, e.g. PRINTERS). | K → adds black color

$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$

subtractive





YIQ (Y = luminance, I = in-phase (orange-blue), Q = quadrature (purple-green))

- ⊕ natural colors & skin — NTSC-US TV standard
- ⊕ compress save channels with use bandwidth

$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ 0{,}596 & -0{,}274 & -0{,}322 \\ 0{,}211 & -0{,}523 & 0{,}312 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$

HSV
L
B

value
lightness
brightness

- ⊕ user oriented color picking
- dimensions no longer primary
- Hue → base color
- Sat → purity

RGB → project to hexagon (not lin. transf)

CODE

RGB → HSV

```
min = min{R,G,B}
max = max
V = max
if (max ≠ 0)   S = (max − min)/max
else           S = 0
H = hue (V, S, R, G, B)
```

Perceptually uniform color spaces

colors close together not necessarily similar visually → Mac Adams (ellipses)

CIE Lab **CIE Luv**

ellipses are now almost circular (nonlinear warps of CIE chart)

***add SB info**

P point A matrix

linear map $A(\alpha x + \beta y) = \alpha Ax + \beta Ay$ — Ax

Rigid $Ax+b$, $A^{-1} = A^T$, $\det A = 1$

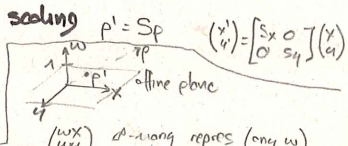
Affine $Ax+b$

2D transformations

transl $p' = p + t$ $\quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$

rotation of θ (z-axis) $\quad p' = Rp$ $\quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

scaling $p' = Sp$ $\quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$



Homogeneous coords

$p = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ affine maps are linear in homog. coord

$p = \begin{pmatrix} wx \\ wy \\ w \end{pmatrix}$ w-along repres (any w) (line in 3D)

$p' = Tp$ $\quad \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$

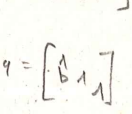
$S/R = \begin{bmatrix} & & 0 \\ & & 0 \\ 0 & 0 & 1 \end{bmatrix}$

shear $SH_x = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}$ → $\quad SH_y = \begin{bmatrix} 1 & 0 \\ b & 1 \end{bmatrix}$ ↑

combining TR_x → rotation followed by transl (order matters)

Commutativity: (✓ T R S; T ✓ ·; R · ✓; S · ✓) $\quad M_1 M_2 = M_2 M_1$ — $n = 2D$ only

$\begin{bmatrix} R & \\ 0 & 1 \end{bmatrix}$ $\begin{bmatrix} I & t \\ 0 & 1 \end{bmatrix}$ $\begin{bmatrix} S & 0 \\ 0 & 1 \end{bmatrix}$



3D transformations

$p = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \rightarrow \begin{pmatrix} x/w \\ y/w \\ z/w \\ 1 \end{pmatrix}$ hyperplane projection

$T = \begin{bmatrix} 1 & & & t_x \\ & 1 & & t_y \\ & & 1 & t_z \\ & & & 1 \end{bmatrix}$ $\quad S = \begin{bmatrix} S_x & & & \\ & S_y & & \\ & & S_z & \\ & & & 1 \end{bmatrix}$

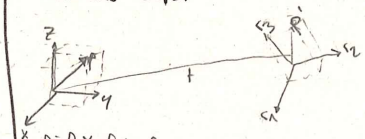
$R_x = \begin{bmatrix} 1 & & & \\ & \cos\theta & -\sin\theta & \\ & \sin & \cos & \\ & & & 1 \end{bmatrix}$ $\quad R_y = \begin{bmatrix} \cos & & \sin & \\ & 1 & & \\ -\sin & & \cos & \\ & & & 1 \end{bmatrix}$ $\quad R_z = \begin{bmatrix} \cos & -\sin & & \\ \sin & \cos & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$ right hand rule

(also represent ccw rotation of axes)

$R(\vec{v}, \theta) = \ldots$ rot around axis $\vec{v}$

$SH_{xy} = \begin{bmatrix} 1 & 0 & sh_x & 0 \\ 1 & sh_y & 0 \\ 1 & 0 \\ & & & 1 \end{bmatrix}$ $\quad SH_{xz} = \begin{bmatrix} 1 & sh_x \\ sh_z & 1 \\ & & 1 \\ & & & 1 \end{bmatrix}$ $\quad SH_{yz} = \begin{bmatrix} 1 & & \\ sh_y & 1 & \\ sh_z & & 1 \\ 0 & & & 1 \end{bmatrix}$

Coordinate syst



$p' = \begin{bmatrix} r_1 & r_2 & r_3 & t \\ 0 & 0 & 0 & 1 \end{bmatrix} p = \begin{pmatrix} t + r_1 p_x + r_2 p_y + r_3 p_z \\ 1 \end{pmatrix}$

TR ≡ change of basis (~~same~~ interpretation)

- ω object
- world
- C camera

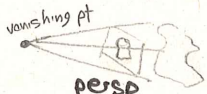
Normals

$p' = Tp \rightarrow n' = (T^{-1})^T n$ $\quad p = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ satisfies $Ax + By + Cz + D = 0$

$n \cdot p = 0$ $\quad n = \begin{pmatrix} A \\ B \\ C \end{pmatrix}$

$(A^{-1})^T = (A^T)^{-1}$ $\quad (Ax)^T = x^T A$

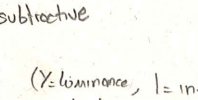
Projection 3D → 2D

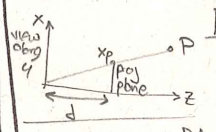


persp

parallel in 3D not anymore in 2D (retina up-close)

parallel





PE

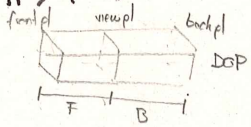
$x_p = dx/z$

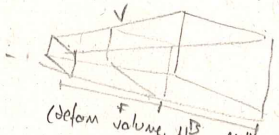
$y_p = dy/z$

$\Pi_{per} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1/d & 0 \end{bmatrix}$ $\quad \Pi_{per} p = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = \begin{pmatrix} xd/z \\ yd/z \\ d \end{pmatrix}$ 3D coord

$\Pi_{orth} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & 1 \end{bmatrix}$ (ignore unwanted coord)

Clipping planes





only objs in volume are drawn

(deform volume, then ~~F~~ B)

Quaternions

complex numbers in 3D (4D)

$z = a_1 + ib + jc + kd$ (1 + vector)

$i^2 = j^2 = k^2 = -1$

$ij = k$ $\quad jk = i$ $\quad ki = j$ $\quad ji = -k$ ...

$ijk = -1$

properties

"=" (coefficients)

sum

$\|z\| = \sqrt{a^2 + b^2 + c^2 + d^2}$

$z = s + \vec{v}$ (scalar + vector)

$\frac{z}{\|z\|}$ unit

$\bar{z} = s - \vec{v}$ (conjugate)

$z_1 z_2 \neq z_2 z_1$

$z\bar{z} = \|z\|^2$ $\quad z^{-1} = \frac{\bar{z}}{\|z\|^2}$

$z z^{-1} = z^{-1} z = 1$

(can be represented as unit length) → infer α from unit vector v

$R(\vec{v}, \theta) = \ldots$ $\quad p \rightarrow$ quat. form $(0 + ix + jy + kz)$

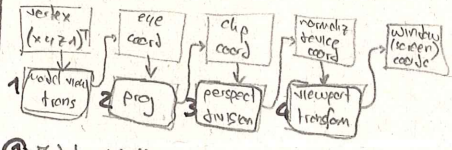
$Q(\vec{v}, \theta) = \underbrace{\cos(\theta/2) + \sin(\theta/2)v}_{q} = q^{-1}$

$p' = q p q^{-1}$ (here p is in quaternion form, pure q unit quaternion)

⊕ easy + efficient

⊕ no gimbal lock

OpenGL transformations



① Model → World

$\begin{bmatrix} r_1 & r_2 & r_3 & t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} wx \\ wy \\ wz \\ w \end{pmatrix} = \begin{pmatrix} wx' \\ wy' \\ wz' \\ w \end{pmatrix}$ $\quad$ World → camera $= \begin{bmatrix} \text{left} & \text{up} & \text{dir} & \text{eye} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} cx \\ cy \\ cz \\ 1 \end{pmatrix}$

left = 100 ᵀ
up = 010 ᵀ
dir = 00-1 ᵀ
eye = 000 ᵀ

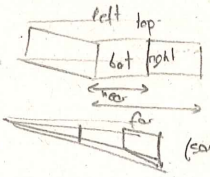
(use inverse matrix)

② Parallel proj | glOrtho (left, right, bottom, top, near, far)

borders

Perspective proj | glFrustum (");

$\begin{bmatrix} cx' \\ cy' \\ cz' \\ 1 \end{bmatrix} = \begin{bmatrix} \text{(2 values skipped)} \\ \text{PZ1} \end{bmatrix} \begin{pmatrix} cx \\ cy \\ cz \\ 1 \end{pmatrix}$



only pts inside are projected

(same)

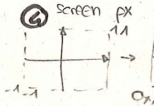
clipping: $p = \begin{pmatrix} cx' \\ \vdots \end{pmatrix}$ compare them with 1 (all?) if smaller = inside frustum

normalized screen device coords

③ $\begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} = \begin{pmatrix} cx'/cz \\ cy'/-cz \\ cz'/+cz \end{pmatrix}$

depth test (dist along neg z)

④ screen px



$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \text{TAT} \\ \text{PZ1} \end{bmatrix}$

glViewport (ox, oy, w, h)

glDepthRange (n, f)

3.1 LIGHTING AND SHADING I

Measuring light = counting photons

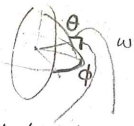
Radiometry -> study of EM radiation, incl light

$\theta = \frac{\ell}{r}$ [rad] (2π tot)

$\Omega = \frac{A}{r^2}$ (4π tot) steradi [sr]

$A = 4\pi r^2$

$V = \frac{4}{3}\pi r^3$



$\omega = (\theta, \phi)$ zenith azimuth

$\omega_x = \sin\theta\cos\phi$

$\omega_y = \sin\theta\sin\phi$

$\omega_z = \cos\theta$

$lat = \frac{2\theta}{\pi}(\pi - \theta)$ $long = \frac{2\theta}{\pi}\phi$

$dA = (r\,d\theta)(r\sin\theta\,d\phi)$

$d\vec{\omega} = \frac{dA}{r^2} = \sin\theta\,d\theta\,d\phi$

$\Omega = \int_{S^2} d\vec{\omega} = \int_0^{2\pi}\int_0^{\pi} \sin\theta\,d\theta\,d\phi = 4\pi$

Photons

$\vec{x}$ pos

$\vec{\omega}$ direction of mot

λ wavel

$E = \frac{hc}{\lambda}$ [J]

$h = 6.63 \cdot 10^{-34}$ [J·kg/s] Plank's const

$c = 299,792.458$ [m/s] (light's speed)

[J = kg·m²/s²]

Radiometry ! *add definitions s8

flux $\Phi(\lambda)$ [W] tot E passing through surface/space per time J/s

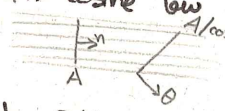
es: photons hitting wall per second / leaving lamp

Irradiance $E(x)$ [W/m²] flux per unit area arriving at surface

$\frac{d\Phi(A)}{dA(x)}$ es: photons hitting wall area/wall area

radiosity $B(x) =$ " es: photons reflecting leaving

Lambert's cosine law (ideal of irradiance / anisotropic bc)



$\mathcal{E} = \frac{\Phi}{A}\cos\theta$ (the more perpendicular, the brighter)

radiant intensity $I(\vec{\omega}) = \frac{d\Phi}{d\vec{\omega}}$ [W/sr] power (flux) per solid angle = directional intensity of flux

$\Phi = \int_{S^2} I(\omega)\,d\vec{\omega}$ es: power per unit solid angle emanating from lamp

$\Phi = 4\pi I$ isotropic pt src (uniform in all directions)

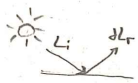
Radiance $L(x,\vec{\omega}) = \frac{dI(\omega)}{dA(x)} = \frac{d^2\Phi(A)}{d\vec{\omega}\,dA(x)\cos\theta}$ [W/m²·sr]

△ raytracing, const along ray (others can be expressed with it)

$dA^{\perp}(x,\vec{\omega})$

Reflection models

BRDF (bidirectional reflectance distrib fct)



differential irradiance -> a cone of directions around $\vec{\omega_i}$

4D (2in 2out) spectral characteristics

$f_r(x,\vec{\omega_i},\vec{\omega_r}) = \frac{dL_r(x,\vec{\omega_r})}{dE_i(x,\vec{\omega_i})}$ [1/sr]

i = incident

r = reflected

complex reflections

anisotropy = property of being reflection dependant

reflection eq (local illum. model)

$L_r(x,\vec{\omega_r}) = \int_{H^2} f_r(x,\omega_i,\omega_r) L_i(x,\vec{\omega_i}) \cos\theta_i\,d\vec{\omega_i}$

SIMPLER REFLECTIONS (phong model) ⊖ more efficient, simpler ⊕ openGL

- **Ambient light** coming from all dir, independent of camera / light / surface orient

$I = I_a k_a$ (light src, material param (const))

- **Diffuse reflection** directed light I_p

dependent on • light pos • surface orientation

indep on • camera (reflected in all dirs)

$I = I_p k_d \cos\theta = I_p k_d (N \cdot L)$

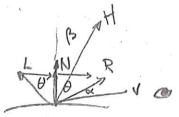
SUM them -> simple model

Attenuation due to spatial radiation

$f_{att} = \frac{1}{d_L^2}$ model in openGL $f_{att} = \min\left(\frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1\right)$

$I = I_a k_a + f_{att} I_p k_d (N \cdot L)$

- **Specular reflection** depends on viewing ray



$R = N\cos\theta + S$

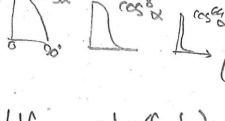
$= 2N\cos\theta - L = 2N(N \cdot L) - L$

$\cos\alpha = R \cdot V = (\ ") \cdot V$

A+D+S

approx specular reflection by cosine powers

$I_\lambda = I_{a_\lambda} k_a O_{d_\lambda} + \sum_{i=1..m} f_{att_i} I_{p_\lambda} [k_d O_{d_\lambda}(N \cdot L_i) + k_s (R_i \cdot V)^{n}]$ (Lorentezzo)



$O_{d_\lambda} = ?$ -> multiple light sources

k_a, k_d, k_s, n material dependant

Halfway vector (faster)

$\cos^n\beta = (N \cdot H)^n$ $H = \frac{L+V}{\|L+V\|}$ (compute for all RGB components)

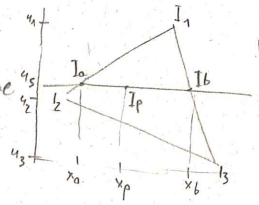
3.2 LIGHTING & SHADING II

Shading models

- flat shading (1 color / primitive)
- gouraud shading (lin. interp of vx intensities)] screen space
- phong " (" normals) ← more sophisticated, (⊖ performance) per px eval
-] obj space

Gouraud

- calc face norm
- calc vx norm (average)
- evaluate ill. mod per vertex
- interpolate vx bilinear on scanline



$I_a = I_1 - (I_1 - I_2)\left(\frac{y_1 - y_s}{y_1 - y_2}\right)$

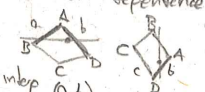
$I_b = (\ " \ )\left(\frac{\ " \ }{\ " \ }\right)$

$I_p = I_b - (I_b - I_a)\left(\frac{x_b - x_p}{x_b - x_a}\right)$

Problems

⊖ perspective distortion

⊖ orientation dependence



⊖ shared vertices

quality dependant on primitive size

Phong barycentric interp of normal

$n_x = \lambda_a n_a + \lambda_b n_b + \lambda_c n_c$

$\lambda_a = \frac{\text{area opposite triangle}}{A_{tot}}$

$\sum\lambda = 1$ $(\lambda_a + \lambda_b + \lambda_c)$

$\lambda_a \cdot a + \lambda_b \cdot b + \lambda_c \cdot c = x$

properties $x = a \rightarrow n_x = n_a$

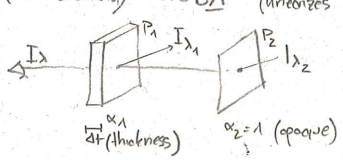
Problems

⊖ normal not defined representative

Transparency

composite transparent obj

Alpha-blending RGBA (linearizes exponential intensity attenuation)



α = absorption

intensity filtered by P_1

- $I_\lambda = \underbrace{I'_{\lambda_1}}_{P_1 \text{ emission}} + I'_{\lambda_2}$

$= I_{\lambda_1}\alpha_1\Delta t + I_{\lambda_2} e^{-\alpha_1\Delta t}$

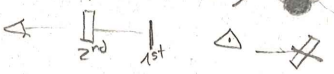
linearize $I'_{\lambda_2} = I_{\lambda_2}(1 - \alpha_1\Delta t)$

$\Rightarrow I_\lambda = I_{\lambda_1}\cdot\alpha_1 + I_{\lambda_2}(1 - \alpha_1)$

Rendering order

needed: sorted polygon traversal

back to front rendering



depth peeling solution

- multiple passes
- each pass renders next closest frag →

Other lighting models • cook-torrance (metal / specular) • oshikh min (anisotropic)

4.1 GEOMETRY & TEXTURES I

0D 1D 2D 3D (manifold)

Applications
- game, movies (mesh)
- engineering (CAD), polynomes
- medicine/biology MRI → 3D volumetric data
- architecture (states)

Sources
real world: scanning, kinect
- point clouds (noisy)

digital: 3D modeling (blend)/vectors
procedural modeling (grammar, rules)

considerations
- storage
- rendering
- acquisition
- creation
- editing

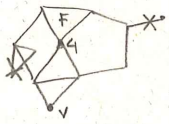
Geometry representation
Polygonal meshes
- boundary
- geometry + connectivity

fast rendering, dedicated GPU pipeline

Polygonal mesh
vertices $v_0, v_1, \dots v_{n-1}$ edges $\{(v_0,v_1),\dots (v_{n-2},v_{n-1})\}$ faces

planar
non-self-intersecting

Mesh = set of connected polygons $M = \langle V, E, F \rangle$



properties
- all edges belong to polygon
- intersection of polygons is - empty - vertex - edge

deg(v)

Boundary: all edges that belong to 1-polygon

Manifold
surface locally homeomorphic to disc
if closed, divides space in 2

not allowed

(real world data often non-manifold)

Mesh DS
(store geometry & topology) + attributes (normal, color, ...)
- vx
- bc
- edges/faces

op: rendering
geometric queries (es vertices of F)
modifications (es remove V)

Triangle list

triangles		
vx ind	vx ind	vx ind
(x_0,y_0,z_0)	(x_1,y_1,z_1)	- -
(, ,)		

⊕ simple
⊖ no connectivity
⊖ redundant
STL file

Indexed face set

triangles	
v ind	v ind
0	1 2
1	2 ...

vertices	
ind	coord
0	(x_0,y_0,z_0)
1	
2	

⊕ efficient
⊕ not-redund
⊕ connectivity
⊖ expensive query + modification

OBJ/OFF/VRL files

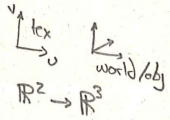
Mesh textures
enhancing realism without increasing geometric complexity

mapping
texture + geometry = rendering

ISSUES
- mapping geom ↔ texture (1:1 map)
- anti-aliasing, filtering
- level of detail (LOD)

es sphere $\begin{bmatrix} u \\ v \end{bmatrix} \to \begin{pmatrix} \sin u \cdot \sin v \\ \cos v \\ \cos u \cdot \sin v \end{pmatrix}$

$x(u,v)$, $y(u,v)$, $z(u,v)$ parametrization



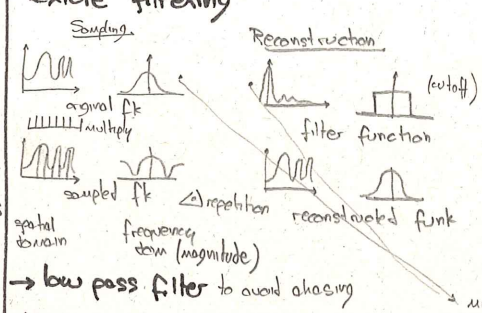
$\mathbb{R}^2 \to \mathbb{R}^3$

desirable properties
- low distortion
- bijective mapping
- efficient comput

finding cuts
texture atlas

4.2 GET II

Texture filtering



→ low pass filter to avoid aliasing

Aliasing
- jaggy edges
- moiré patterns
- loss of detail

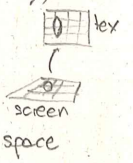
△ high freq

structured vs unstructured noise
↳ we like it better (retina discretizes light, but very well → no Aliasing)

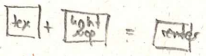
isotropic 2D gaussian $f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$

anisotropic $= \frac{1}{2\pi\sigma_x\sigma_y} e^{-\left(\frac{x^2}{2\sigma_x^2}+\frac{y^2}{2\sigma_y^2}\right)}$

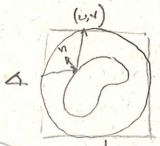
(spatially-varying tex filter)



Light maps
- simulate effect of local light src
- ⊕ precomputed / dyn. adapted



Environment maps
- rendering reflective obj efficiently
- normal-color correspondence

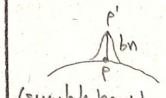


△ tex coor computed intersecting reflected ray with sphere

(use photos per texture)

↳ cube allows simpler computation

Bump mapping
- perturb surface normal according to tex
- small scale geometry ⇒ positional perturbation

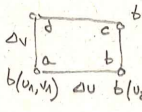


(simulate height map)

$n = \frac{\partial p}{\partial u} \times \frac{\partial p}{\partial v} = p_u \times p_v$ u,v parameter domain

$p' = p + \frac{bn}{|n|}$ $n' = \frac{\partial p'}{\partial u} \times \frac{\partial p'}{\partial v}$ $\frac{\partial p'}{\partial u} = \frac{\partial p}{\partial u} + \frac{\partial (bn)}{\partial u |n|}$

$n' = n + b_v (n \times p_u) + b_u (n \times p_v)$ (uses FD method)



$b_u = \frac{b-a+c-d}{2\Delta u}$

$b_v = \frac{d-a+c-b}{2\Delta v}$

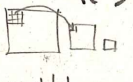
⊖ no silhouette bumps
⊖ no self occlusion
" shadows

Displacement mapping
- perturb geometry based on texture

Procedural/solid textures
- perlin noise
- gabor noise

(⊕ control over spectral prop)

Mip-mapping
- store down-sampled version of tex
- use low-res for far-away obj
- interpolate in between



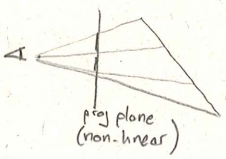
weighted average for next px

⊕ no aliasing
⊕ efficient

Perspective interpolation
tex coord of v_x → tex coord of px
- linear interp in screen space

wanted | we get

Problem: non-linear variation in screen coords



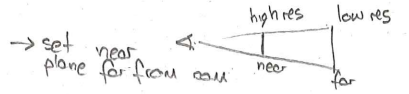
5.1 VISIBILITY & SHADOWS

Visibility
problem: some parts of some surfaces are occluded

Painter's alg solution
render objs/polygons from furthest to nearest
⊖ cyclic overlaps
⊖ intersections

z-buffering store z-value ∀px init to ∞
∀polygon: if z value smaller for one pixel, update it

⊖ limited res & non-linear resolution
→ set near plane far from cam



Shadows
- depth cue
- scene lighting (es point/area light)
- realism

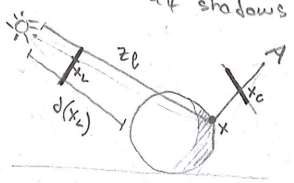
1 - planar shadows
- draw projection of obj on ground

⊖ no self shadows / on other obj
⊖ curved surfaces not supported

2 - projective texture shadows
- separate obstacle / receiver
- compute b/w image from light → use it as projective texture

3 - shadow maps (high end)
- compute depths from light camera

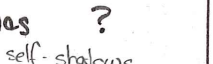
⊖ specific obs/receiver
⊖ no self shadows



∀px on camera plane
- find x in world coord
- project onto light plane (x_L)
- compare $d(x_L)$ shadow map & z_L (dist)

$d(x_L) < z_L \Rightarrow$ x in shadow

Bias ?
avoid self-shadows → add bias (hard to choose)
$d(x_L) + bias < z_L$
es. too much / little

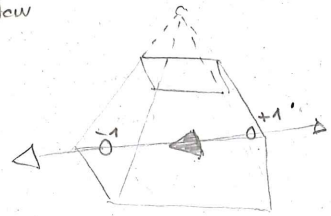


⊖ FOV (field of view)
es points outside light's fov
→ use CUBICAL shadow map

⊖ aliasing (undersampling)
sol: • don't filter depth
• filter result of test → produces fake soft shadow

4 - shadow volumes
- explicitly represent volume of space in shadow
- if polygon inside → shadow
- similar to clipping
- naive $O(\#pol \times \#lights)$

shoot ray from eye
each time boundary intersected → c++ / c--
if c>0 → primitive in shadow
=0 → primitive in light



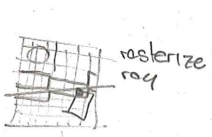
⊕ optimization: use silhouette edges only (back & front-facing west of polygons)
⊖ a bit of geometry
⊖ expensive (skinny triangles)
⊖ watertight silhouette

Acceleration
- Complex scene 600 years
- acceleration: fewer intersections (Mmm)

cost = $O(\#rays \times \#obj)$

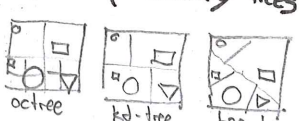
uniform grids
- compute BB
- set grid res
- store references

→ only check when possible intersection



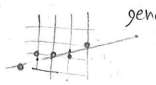
⊕ fast to build
⊕ easy to code
⊖ non-adaptive to scene

space partitioning trees



5.1B SCAN CONVERSION (rasterization)

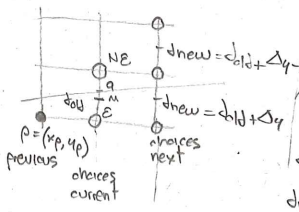
Lines
generate discrete px values



Bresenham line (choose closest px at each intersection) ⊕ fast decision of next p

Criterion: midpoint M
intersection q

line: $f(x,y) = ax + by + c = 0$ (implicit form)
$d = f(M) = f(x_p + 1, y_p + \frac{1}{2})$ $>0 \rightarrow$ NE
$<0 \rightarrow$ E (select)



E	NE
$d_{old} = f(x_p + 1, y_p + \frac{1}{2})$	$d_{old} = 11$
$d_{new} = f(x_p + 2, y_p + \frac{1}{2})$	$d_{new} = f(x_p + 2, y_p + \frac{3}{2})$
$d_{new} - d_{old} = a = \Delta y$	$d_{new} - d_{old} = a + b = \Delta y - \Delta x$

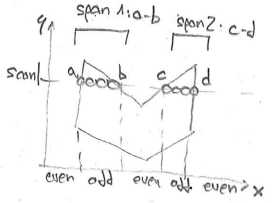
→

$a = \Delta y$
$b = -\Delta x$
?

Polygons
filled polygons, △ very important
(GPU con △s)
- spatial coherence
  - naive: inside test ∀px inefficient
  - scanline

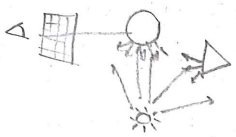
span = group of px inside scanline

Alg: .all scanline inters
.sort them by x
• fill spans if odd parity

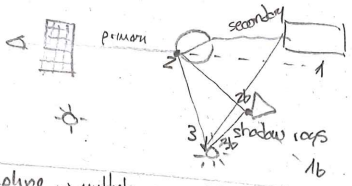


5.2 RAY TRACING
- send rays into scene to determine fx color

Forward



Backward



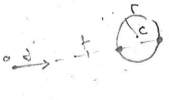
pipeline
1. ray generation
2. intersection
3. shading

Ray generation

super sampling → multiple rays per fx

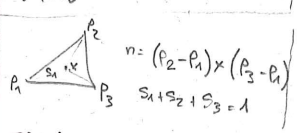
Intersection

Ray $r(t) = \vec{o} + t\vec{d}$



Sphere $\|x - c\|^2 - r^2 = 0$
$\|o + td - c\| \dots$ (x = point of interest)

Triangle barycentric $x = s_1 p_1 + s_2 p_2 + s_3 p_3$



$n = (p_2 - p_1) \times (p_3 - p_1)$
$s_1 + s_2 + s_3 = 1$

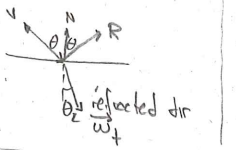
intersect plane
$(o + td - p_1) \cdot n = 0 \rightarrow t = -\frac{(o - p_1) \cdot n}{d \cdot n}$

compute s_i (check $s_i \in [0,1]$)

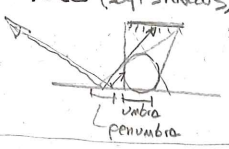
Shading too costly be physically accurate
simplify: surf. reflectance (diff / spec / amb / transparency)
shadows (shadow rays for point in shadow / not)

Extensions

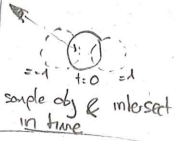
refraction



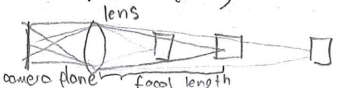
multiple lights / area (soft shadows, multiple shadow rays)

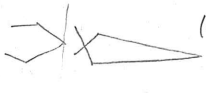


motion blur



Depth of field (DOF)





(out of focus)

7.1 ANIMATION

Animation → bringing images to life

what
- characters
- faces
- hair
- elastic materials
- natural phenomena
- cloth
- herds (fish)
- crowds

How
- keyframing
- skeletal (rig), character
- subspace param
- pbs, particle systems
- motion capture
- video based (facial
- example based (herds)
- procedural

Considerations
- control
- efficiency
- realism / style

- fast deformation with intuitive controls

controllers: classical, sketch-based, keyframing

Animation pipeline
1) story-boarding (planning)
2) concept design (sketches character/environm, features)
3) 3D modeling (geometry & textures)
4) rigging (embed) animation controllers/skeleton/ ready to keyframe)
5) blend-shapes creation (facial expression / generation using blending)
6) animation (key-frames for controllers / interpolation)
7) post-effects (lighting/shading/render

↳ linear space $\sum w_i f_i$ force exp

Rigging
attach skeleton to model using bones (embed).
↳ keyframe to animate

Bone, joint (hierarchy) — rigid transformation

R_i t_i

Weights: indicate how much a vertex is influenced by bone

T = transform (homogeneous coords)

$T(x) = avg(T_1, T_2, w_1, w_2)$

$= w_1(x) T_1 + w_2(x) T_2$

$x' = T(x)x$

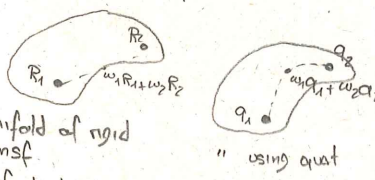
transl: $w_1 t_1 + w_2 t_2$

rotation: $w_1 R_1 + w_2 R_2$ invalid rot matrix!

valid: $R^T = R^{-1}$, $\det(R) = 1$

linear blend skinning
how to blend transf
⊖ problems: elbow

twist



Manifold of rigid transf

(manif: topological space that resembles euclidian space)

" using quat

spherical blend skinning
blend using $(q_2 q_1^{*})^{\omega_2} q_1$

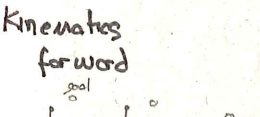
for more quat, a good approx is $\sum_i \omega_i q_i$

since $R \to q$ is not linear, also interpolation is not → interp. curve

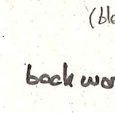
Rigid motion with dual quaternions (8 params - $2^{\text{constr}} = 6$, like $R+t$)

(blending)

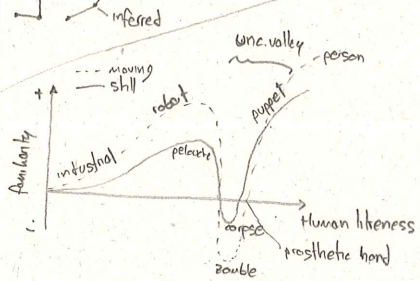
Kinematics
forward



backward



Facial animation
uncanny valley



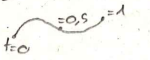
△ fine-scale details → capture

7.2 GEOMETRY PROCESSING

Parametric curves & surfaces $f: X \subseteq \mathbb{R}^m \to Y \subseteq \mathbb{R}^n$

curves 1D → 2D

$s(t) = (x(t), y(t))$



circle $p(t) = r(\cos t, \sin t)$ $t \in [0, 2\pi[$

Bezier

$s(t) = \sum_{i=0}^{n} p_i B_i^n(t)$

$= \binom{n}{i} t^i (1-t)^{n-i}$

1D → 3D (curves in 3D) $s(t) = (x(t), y(t), z(t))$

surfaces 2D → 3D $s(u,v) = (x(u,v), y, z)$

sphere: $s(u,v) = r \begin{pmatrix} \cos u \cos v \\ \sin u \cos v \\ \sin v \end{pmatrix}$ $(u,v) \in [0, 2\pi[\times [-\frac{\pi}{2}, \frac{\pi}{2}]$

Bezier $s(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} p_{ij} B_i^m(u) B_j^n(v)$

Normal: $s_u = \frac{\partial s(u,v)}{\partial u}$ $s_v = \frac{\partial s(u,v)}{\partial v}$ $n = (s_u \times s_v).\text{norm}$

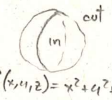
Volumetric repr $3D \to 1D$ (color) (each point in space has color) MRI medic scan

⊕ easy to generate
⊕ point-wise properties
⊕ control by hand

⊖ hard determine in/on/out
⊖ reverse engineering

Implicit curves & surfaces $f: \mathbb{R}^m \to \mathbb{R}$

$S = \{x \in \mathbb{R}^2 \mid f(x) = 0\}$ $S = \{x \in \mathbb{R}^3 \mid \dots\}$



$f(x) > 0$ out
$= 0$ on
< 0 in

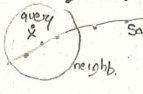
$f(x,y,z) = x^2 + y^2 + z^2 - r^2$

Surface normal $\nabla f(x,y,z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$

⊕ easy determine in/on/out
⊕ easy to combine
⊖ hard to generate points
⊖ limited set
⊖ no real time

Point set surfaces
(scans) acquired data
- point-wise attributes

local fitting approach



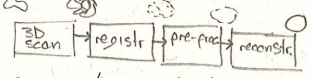
- implicit representation & fast proj.
- find line for each neighborhood

⊕ easy in/on/out
⊕ easy generate points
⊕ reconstruction real data
⊕ real time

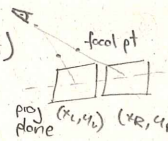
⊖ modeling

⊕ robust to noise (mesh conversion)
⊕ direct rendering
↳ in space, octree tree with grid (dividing quad/tri)

Shape acquisition (digitalization of real world)



1. 3D scanning (touch+ optical)
   - active
   - passive (no control over light)



2. registration → all scans in same coord syst (different views)
   - iterative closest pt alg
   - feature based methods (descriptors)

repeat (rigid motion)

3. pre-processing (cleaning / repairing / resampling) uniform

4. reconstruction → mathematical repres. of shape
   - VORONOI → connect points
   - APPROX → efficient, robust

Modeling tools / Editing geometry
interactive / sketch based UI
deformations (free / elastic / skeletal / struct aware)
cutting / fracturing
smoothing / filtering
compression / simplific / subdivision surfaces
differential properties / segmentation
symmetry / structure detection / feature extraction
saliency

CG - OPEN GL code

```
void main() {
    float r = 1.0;   //normalized [0..1]   8bits/comp = 256color
    g...
gl_FragColor = vec4(r,g,b,a);   //rgba and fx stolen
}
```

ADD CG

1.2 vx shad
p9-11 fr shad

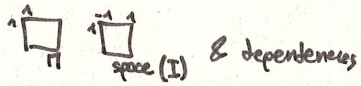
4.1 • OBJ files
(p34) • ~~texture~~ + load
- bind
- lookup

VC PCA steps

focus (understand) Lucas-Kanade

Questions VC

p35 corel/conv indices
p49 infinite support



& dependeneces

Questions CG

clipping stopped ?
gave chapter ?

3.1 O_{d_2} ?

5.1 shad-map bias ?

5.1b $a = \Delta y$ $b = -\Delta x$

7.1 dual quaternions?